Interfaces

Programação Orientada a Objeto II

O que são interfaces?

A interface é um recurso muito utilizado em Java e outras linguagens orientadas a objeto. Em uma interface definimos métodos que uma classe que implementar aquela interface deverá cumprir.

Em Java, uma interface não é uma classe, mas um conjunto de requisitos para classes que precisam adequar-se a ela.

Assim, vamos pensar em **interfaces como contratos** onde o foco está em "**o quê**" deve ser feito, mas não "**como**".

O que são interfaces?

Nota: *Em Java podemos definir herança apenas de uma superclasse, enquanto que podemos implementar várias interfaces.*

Para entender melhor sobre interfaces e contratos, vamos adicionar à classe **Empregado** uma funcionalidade de comparação para poder realizar ordenações de forma prática.

Nosso critério de ordenação serão as matrículas.

Sintaxe

Vamos analisar a interface java.lang.Comparable pertencente à linguagem Java.

```
public interface Comparable {

    int compareTo (Object other);

}
```

```java
public class Empregado implements Comparable {
    private int matricula;
    //...
    @Override
    public int compareTo(Object other) {
        Empregado empregado = (Empregado) other;
        if (this.matricula < empregado.matricula) {
            return -1;
        }
        if (this.matricula > empregado.matricula) {
            return 1;
        }
        return 0;
    }
}
```

Voltando à implementação do método compareTo, veja que definimos a comparação entre objetos **Empregado** através de comparação dos atributos matricula.

Note que este é um método de objeto, ou seja, estamos comparando este objeto e outro. Por exemplo:

```
Empregado joao = new Empregado();
Empregado maria = new Empregado();

int result = joao.compareTo(maria);
```

A interface java.util.Collection

Até agora, trabalhamos com a estrutura de dados array para representar grupos de valores semelhantes em uma mesma variável. No entanto, não é prático ficar manipulando arrays diretamente.

Java fornece uma ampla e poderosa estrutura de coleções através da API Java Collection. Ainda não é o momento de estuda-la em detalhes.

Mas, de forma resumida, temos a interface `java.util.List` que representa as estruturas de lista, e a sua classe mais popular é a `java.util.ArrayList`, que implementa o contrato definido em List usando estruturas de arrays.

show me
Collection.sort Method

Interface de Marcação

Interface de marcação é um recurso utilizado para quando temos interfaces que não possuem métodos ou campos.

A implementação dessa interface em um classe serve para marcar que uma classe pertence à categoria da interface implementada e indicar à JVM que aquela classe terá um comportamento de acordo com o contrato definido.

Classe Abstrata vs Interface

Interface vs Classes Abstratas

Interfaces

No **Java 8+**, tanto as *classes abstratas* quanto as *interfaces* permitem definir contratos para outras classes, mas há diferenças importantes na forma como isso é feito.

Principais diferenças:

Característica	**Classe Abstrata**	**Interface (Java 8+)**
Métodos abstratos	sim	sim
Métodos concretos	sim	sim (default e static)
Atributos	sim (qualquer modificador)	apenas public static final (constantes)
Construtores	sim	não
Herança múltipla	não (só pode estender 1 classe)	sim (pode implementar várias interfaces)
Modificadores de método	public, protected, private	public (por padrão), default, static

Classe Abstrata

Interfaces

Uma classe abstrata pode conter **métodos abstratos** (sem implementação) e **métodos concretos** (com implementação).

Ela pode ter atributos e construtores, e serve como um modelo para outras classes.

Características:

- Pode ter métodos abstratos e concretos.
- Pode ter atributos e construtores.
- Suporta herança única (**uma classe só pode herdar de uma classe abstrata**).

Interface

Interfaces

Uma interface define apenas comportamentos (métodos), mas a partir do java 8+ os métodos podem conter implementação.

Uma classe pode implementar múltiplas interfaces.

Características:

- Métodos abstratos, default (com implementação) e static.
- Sem atributos de instância, apenas constantes public static final.
- Sem construtores.
- Pode ser implementada por várias classes (herança múltipla).

Exemplo Código
Classe Abstrata + Interface

Exemplo

Classe abstrata e Interface

Cenário:

Vamos modelar uma aplicação onde temos **veículos aquáticos** e **aéreos**. Alguns veículos podem **navegar** (como barcos), enquanto outros podem **voar** (como aviões).

- A classe abstrata **Veiculo** fornecerá um comportamento comum para todos os veículos, como um nome e um método para exibir informações.
- As interfaces **Navegavel** e **Voavel** definirão comportamentos específicos para veículos que podem navegar e voar, respectivamente.

Exercícios

Exercício 1: Cadastro e Listagem de Clientes

Cenário: Uma loja virtual precisa cadastrar clientes e listar seus dados.

Dados de Entrada:

- Nome
- Email
- Tipo do cliente (Pessoa Física ou Jurídica)

O que Manipular?

- Criar um modelo que suporta múltiplos tipos de clientes sem alterar a estrutura principal.
- Cada tipo de cliente deve ter comportamentos distintos, como exibição formatada do CPF/CNPJ.
- Listar todos os clientes cadastrados utilizando um método funcional.

Exercício 2: Processamento de Pedidos e Cálculo de Total

Cenário: Um cliente faz uma compra no marketplace e adicionar produtos ao carrinho.

Dados de Entrada:

- Lista de produtos adicionados ao carrinho (Nome, Quantidade, Preço Unitário).

O que Manipular?

- Criar um modelo que suporta múltiplos tipos de produtos.
- Calcular o valor total do pedido iterando sobre os itens do carrinho.
- Permitir a aplicação de descontos em determinados produtos.

Exercício 3: Escolha de Métodos de Pagamento

Cenário: O cliente precisa escolher como deseja pagar a compra.

Dados de Entrada:

- Valor total da compra.
- Método de pagamento escolhido (Cartão de Crédito, Boleto, Pix).

O que Manipular?

- Criar um sistema que aceite diferentes métodos de pagamento sem precisar alterar a lógica de compra.
- Processar o pagamento conforme o método escolhido.
- Exibir mensagens distintas dependendo da forma de pagamento.

Exercício 4: Rastreamento de Entrega do Pedido

Cenário: O cliente acompanhar o status do pedido após a compra.

Dados de Entrada:

- Código do pedido.
- Status da entrega (Processando, Enviado, Entregue).

O que Manipular?

- Criar um sistema que gerencie e exiba diferentes tipos de entrega.
- Simular a atualização automática do status do pedido.
- Notificar o cliente sobre mudanças no status utilizando uma abordagem funcional.

Exercício 5: Envio de Notificações de Compra

Cenário: O cliente recebe notificações sobre a compra realizada.

Dados de Entrada:

- Tipo de notificação desejada (Email, SMS, Push Notification).
- Mensagem sobre o status da compra.

O que Manipular?

- Criar um sistema que envie notificações utilizando diferentes canais.
- Garantir que novas formas de notificação possam ser adicionadas sem modificar a lógica principal.
- Processar o envio das notificações iterando sobre a lista de clientes.

Obrigada